Contents

Trust Domain Migration

4 minute read ✔ page test

This task shows you how to migrate from one trust domain to another without changing authorization policy.

In Istio 1.4, we introduce an alpha feature to support trust domain migration for authorization policy. This means if an Istio mesh needs to change its trust domain, the authorization policy doesn’t need to be changed manually. In Istio, if a workload is running in namespace `foo` with the service account `bar`, and the trust domain of the system is `my-td`, the identity of said workload is `spiffe://my-td/ns/foo/sa/bar`. By default, the Istio mesh trust domain is `cluster.local`, unless you specify it during the installation.

Before you begin

Before you begin this task, do the following:

1. Read the Istio authorization concepts.
2. Install Istio with a custom trust domain and mutual TLS enabled.

    ```
    $ istioctl install --set profile=demo --set meshConfig.trustDomain=old
    -td
    ```

3. Deploy the httpbin sample in the `default` namespace and the sleep sample in the `default` and `sleep-allow` namespaces:

```
$ kubectl label namespace default istio-injection=enabled
$ kubectl apply -f @samples/httpbin/httpbin.yaml@
$ kubectl apply -f @samples/sleep/sleep.yaml@
$ kubectl create namespace sleep-allow
$ kubectl label namespace sleep-allow istio-injection=enabled
$ kubectl apply -f @samples/sleep/sleep.yaml@ -n sleep-allow
```

4. Apply the authorization policy below to deny all requests to `httpbin` except from `sleep` in the `sleep-allow` namespace.

```
$ kubectl apply -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
  name: service-httpbin.default.svc.cluster.local
  namespace: default
spec:
  rules:
  - from:
    - source:
        principals:
        - old-td/ns/sleep-allow/sa/sleep
    to:
    - operation:
        methods:
        - GET
  selector:
    matchLabels:
      app: httpbin
---
EOF
```

Notice that it may take tens of seconds for the authorization policy to be propagated to the sidecars.

1. Verify that requests to `httpbin` from:
    - `sleep` in the `default` namespace are denied.

```
$ kubectl exec "$(kubectl get pod -l app=sleep -o jsonpath={.items..me
tadata.name})" -c sleep -- curl http://httpbin.default:8000/ip -sS -o
/dev/null -w "%{http_code}\n"
403
```

    - `sleep` in the `sleep-allow` namespace are allowed.

```
$ kubectl exec "$(kubectl -n sleep-allow get pod -l app=sleep -o jsonp
ath={.items..metadata.name})" -c sleep -n sleep-allow -- curl http://h
ttpbin.default:8000/ip -sS -o /dev/null -w "%{http_code}\n"
200
```

Migrate trust domain without trust domain aliases

1. Install Istio with a new trust domain.

   ```
   $ istioctl install --set profile=demo --set meshConfig.trustDomain=new
   -td
   ```

2. Redeploy istiod to pick up the trust domain changes.

   ```
   $ kubectl rollout restart deployment -n istio-system istiod
   ```

   Istio mesh is now running with a new trust domain, `new-td`.

3. Redeploy the `httpbin` and `sleep` applications to pick up changes from the new Istio control plane.

```
$ kubectl delete pod --all
```

```
$ kubectl delete pod --all -n sleep-allow
```

4. Verify that requests to `httpbin` from both `sleep` in `default` namespace and `sleep-allow` namespace are denied.

```
$ kubectl exec "$(kubectl get pod -l app=sleep -o jsonpath={.items..me
tadata.name})" -c sleep -- curl http://httpbin.default:8000/ip -sS -o
/dev/null -w "%{http_code}\n"
403
```

```
$ kubectl exec "$(kubectl -n sleep-allow get pod -l app=sleep -o jsonp
ath={.items..metadata.name})" -c sleep -n sleep-allow -- curl http://h
ttpbin.default:8000/ip -sS -o /dev/null -w "%{http_code}\n"
403
```

This is because we specified an authorization policy that

deny all requests to `httpbin`, except the ones the `old-td/ns/sleep-allow/sa/sleep` identity, which is the old identity of the `sleep` application in `sleep-allow` namespace. When we migrated to a new trust domain above, i.e. `new-td`, the identity of this `sleep` application is now `new-td/ns/sleep-allow/sa/sleep`, which is not the same as `old-td/ns/sleep-allow/sa/sleep`. Therefore, requests from the `sleep` application in `sleep-allow` namespace to `httpbin` were allowed before are now being denied. Prior to Istio 1.4, the only way to make this work is to change the authorization policy manually. In Istio 1.4, we introduce an easy way, as shown below.

Migrate trust domain with trust domain aliases

1. Install Istio with a new trust domain and trust domain aliases.

```
$ cat <<EOF > ./td-installation.yaml
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
spec:
  meshConfig:
    trustDomain: new-td
    trustDomainAliases:
      - old-td
EOF
$ istioctl install --set profile=demo -f td-installation.yaml -y
```

2. Without changing the authorization policy, verify that requests to `httpbin` from:

   - `sleep` in the `default` namespace are denied.

```
$ kubectl exec "$(kubectl get pod -l app=sleep -o jsonpath={.items..me
tadata.name})" -c sleep -- curl http://httpbin.default:8000/ip -sS -o
/dev/null -w "%{http_code}\n"
403
```

   - `sleep` in the `sleep-allow` namespace are allowed.

```
$ kubectl exec "$(kubectl -n sleep-allow get pod -l app=sleep -o jsonp
ath={.items..metadata.name})" -c sleep -n sleep-allow -- curl http://h
ttpbin.default:8000/ip -sS -o /dev/null -w "%{http_code}\n"
200
```

Best practices

Starting from Istio 1.4, when writing authorization policy, you should consider using the value `cluster.local` as the trust domain part in the policy. For example, instead of `old-td/ns/sleep-allow/sa/sleep`, it should be `cluster.local/ns/sleep-allow/sa/sleep`. Notice that in this case, `cluster.local` is not the Istio mesh trust domain (the trust domain is still `old-td`). However, in authorization policy, `cluster.local` is a pointer that points to the current trust domain, i.e. `old-td` (and later `new-td`), as well as its aliases. By using `cluster.local` in the authorization policy, when you migrate to a new trust domain, Istio will detect this and treat the new trust domain as the old trust domain without you having to include the aliases.

Clean up

```
$ kubectl delete authorizationpolicy service-httpbin.default.svc.cluster.lo
cal
$ kubectl delete deploy httpbin; kubectl delete service httpbin; kubectl de
lete serviceaccount httpbin
$ kubectl delete deploy sleep; kubectl delete service sleep; kubectl delete
 serviceaccount sleep
$ istioctl x uninstall --purge
$ kubectl delete namespace sleep-allow istio-system
$ rm ./td-installation.yaml
```